PostgreSQL 原理简介

GALY LEE
galylee@gmail.com 李元佳

PostgreSQL Conference China, 2011@Guangzhou

内容

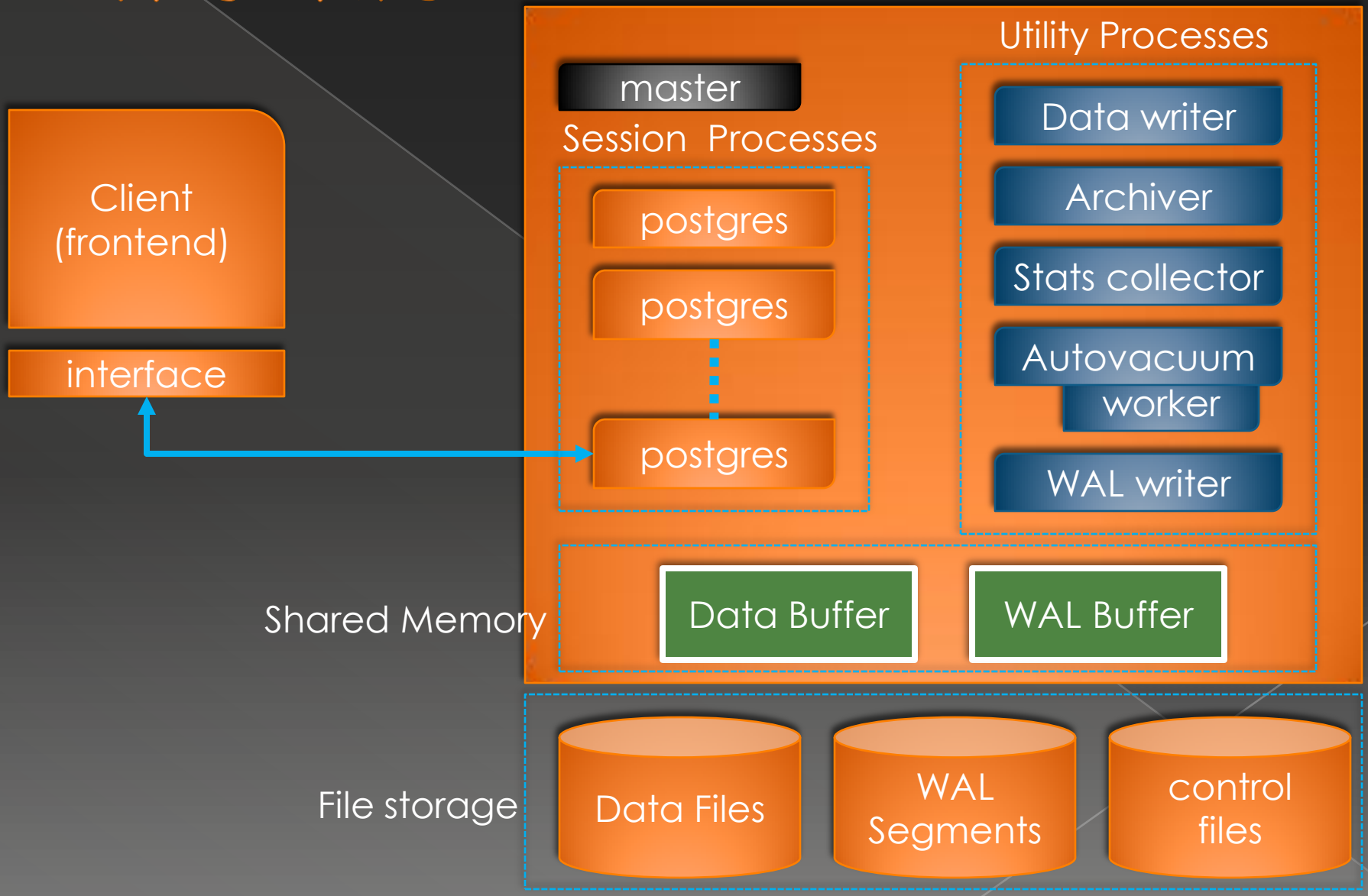
- 体系架构
- 进程结构
- 内存结构
- 存储结构
- 多版本并发控制MVCC
- 预写式事务日志WAL
- Postgres的架构

概要

- 面向运维人员的介绍
- 技术要点作为重点介绍
- 以理解概念为主
- 不涉及PG内部机制

体系架构

体系架构


Client
(frontend)
interface
master
Session Processes
postgres
postgres
postgres
Utility Processes
Data writer
Archiver
Stats collector
Autovacuum
worker
WAL writer
Shared Memory
Data Buffer
WAL Buffer
File storage
Data Files
WAL
Segments
control
files

进程结构

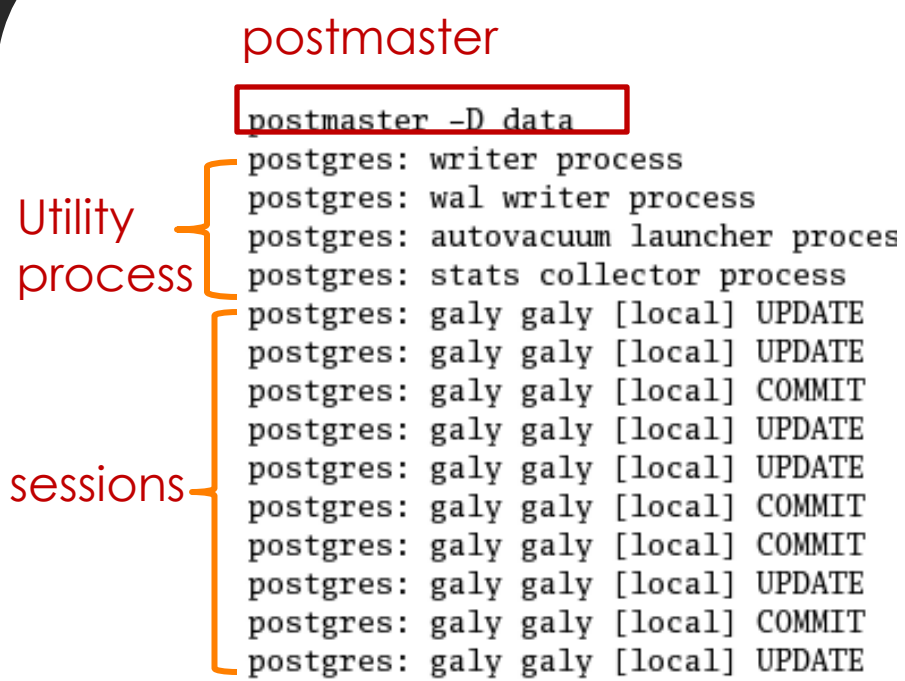
进程

- process-per-user
  - 一个用户具有一个单独的进程
- Why Process
  - 线程库支持的复杂性
    - 在各个操作系统平台下线程库具有很大的差异性
  - 健壮性
    - 单个进程的崩溃，不会导致数据库的崩溃
  - 复杂性
    - 多线程的一些竞态极难调试
  - 在LINUX及BSD下，进程与线程的开销区别并不大

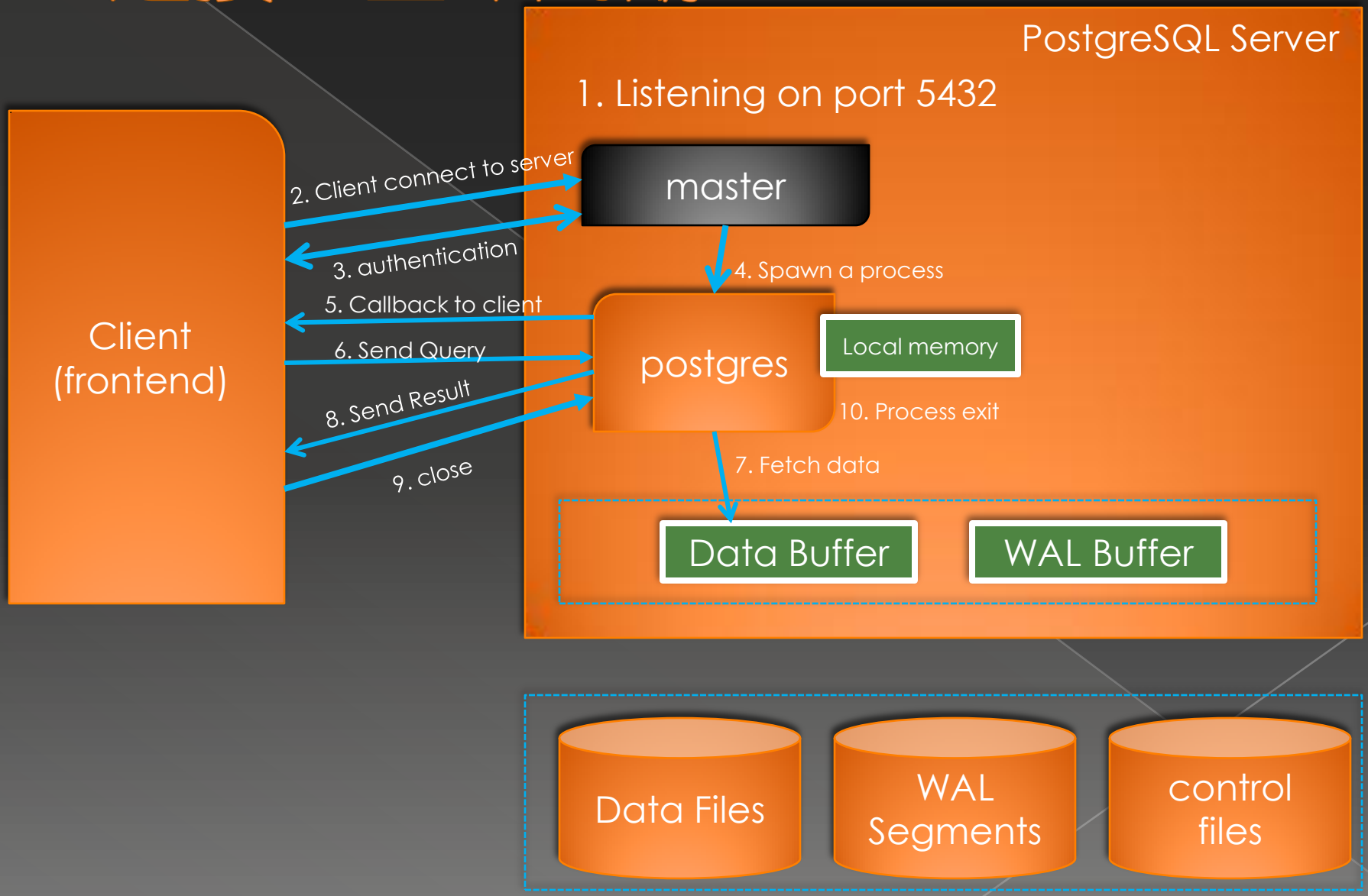
进程结构

- Postmaster
  - Supervisory daemon process waiting for requests to databases.
  - Starts a backend process each time it receives a request.
  - Multiple postmasters can run on a single machine, as long as each one is running on a different port.
- Postgres(backend)
  - Executes the SQL queries.
  - Separate backend process for each query currently being executed.
- Data Writer
- Wal writer
- Autovacuum launcher
- Autovacuum worker
- Stats collector

PostgreSQL Run Time Process List

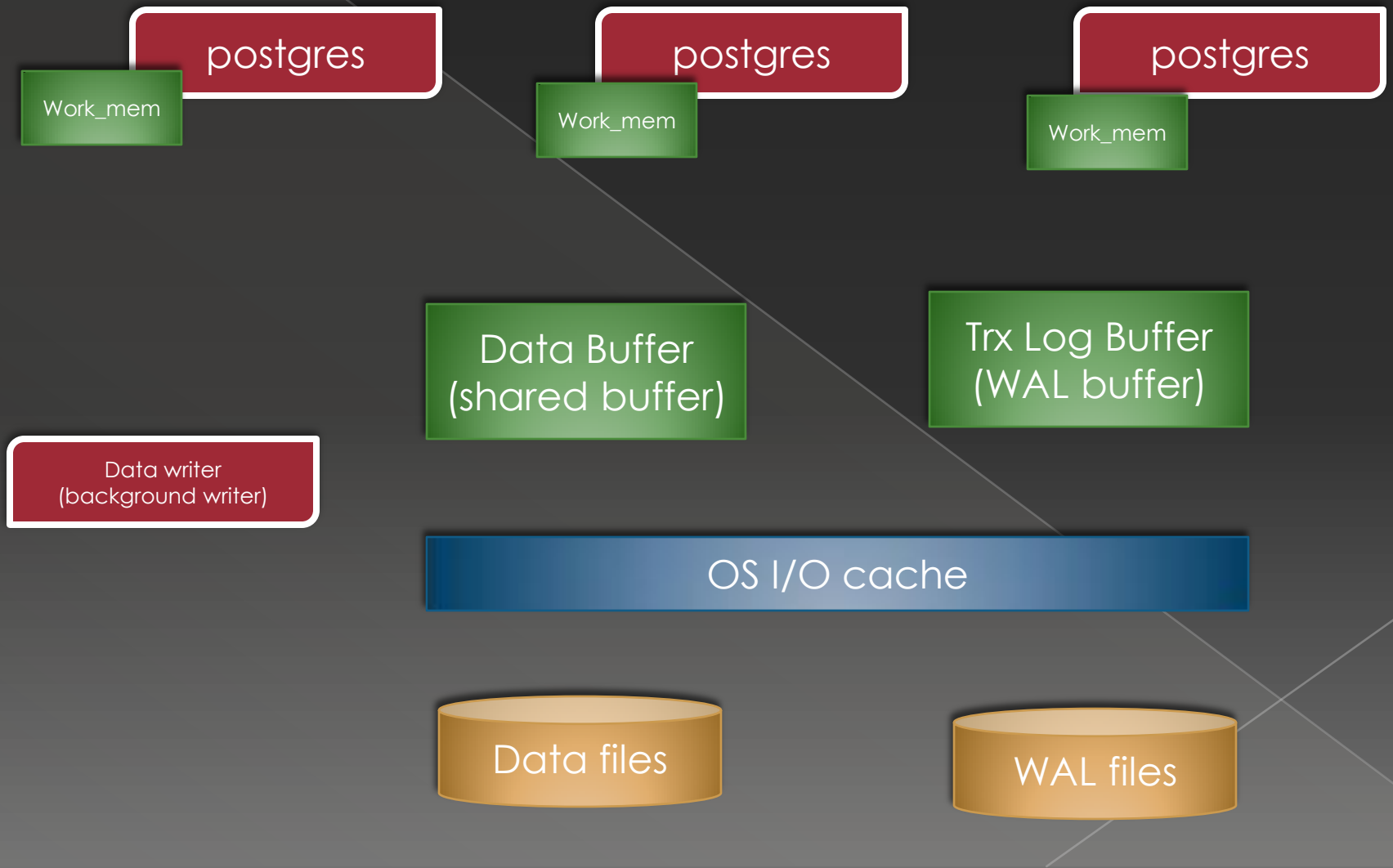


连接的生命周期

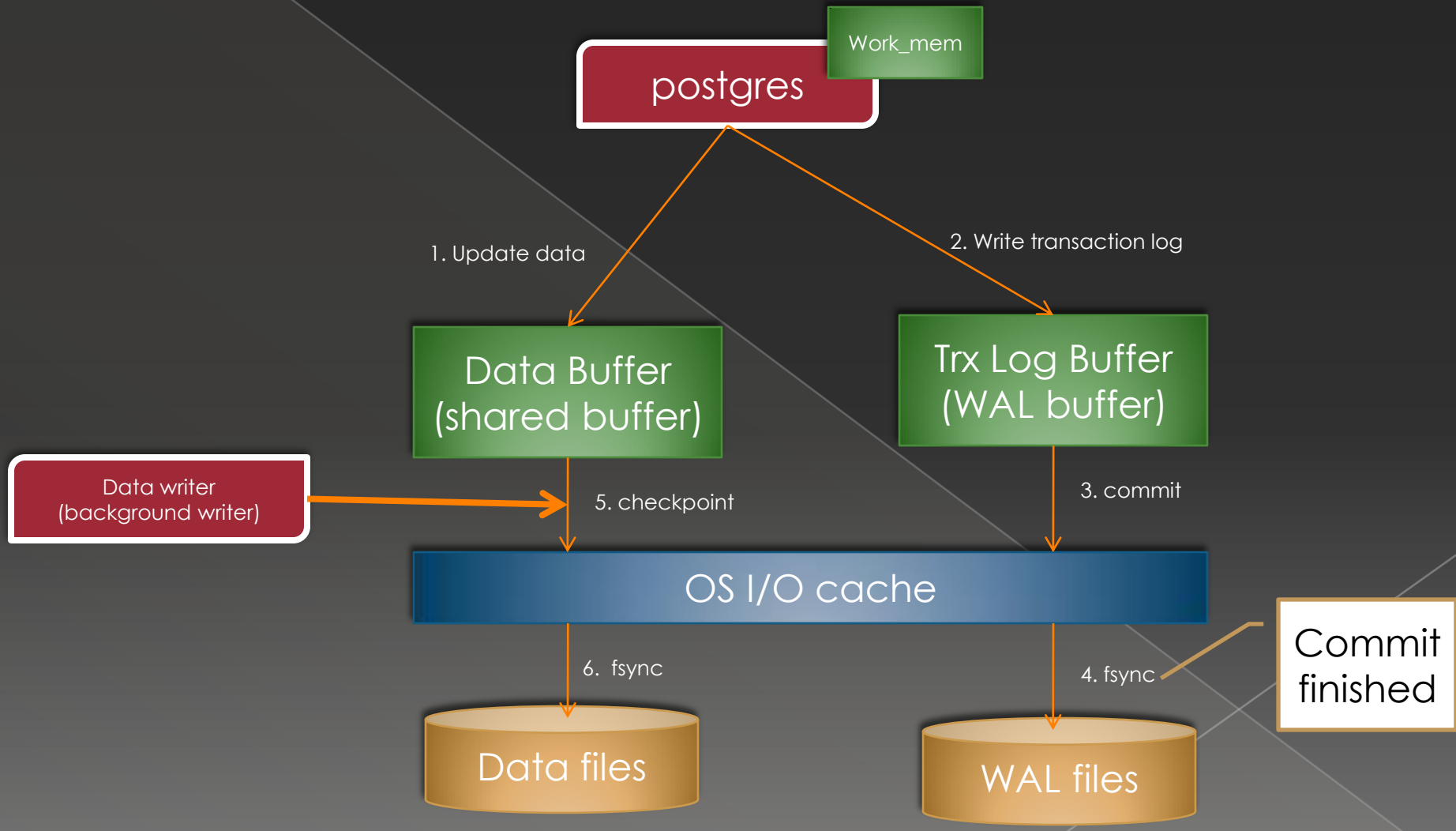

PostgreSQL Server
1. Listening on port 5432
Client
(frontend)
2. Client connect to server
3. authentication
4. Spawn a process
5. Callback to client
6. Send Query
7. Fetch data
8. Send Result
9. close
10. Process exit
master
postgres
Local memory
Data Buffer
WAL Buffer
Data Files
WAL
Segments
control
files

内存结构

内存的结构


postgres
Work_mem
postgres
Work_mem
postgres
Work_mem
Data Buffer
(shared buffer)
Trx Log Buffer
(WAL buffer)
Data writer
(background writer)
OS I/O cache
Data files
WAL files

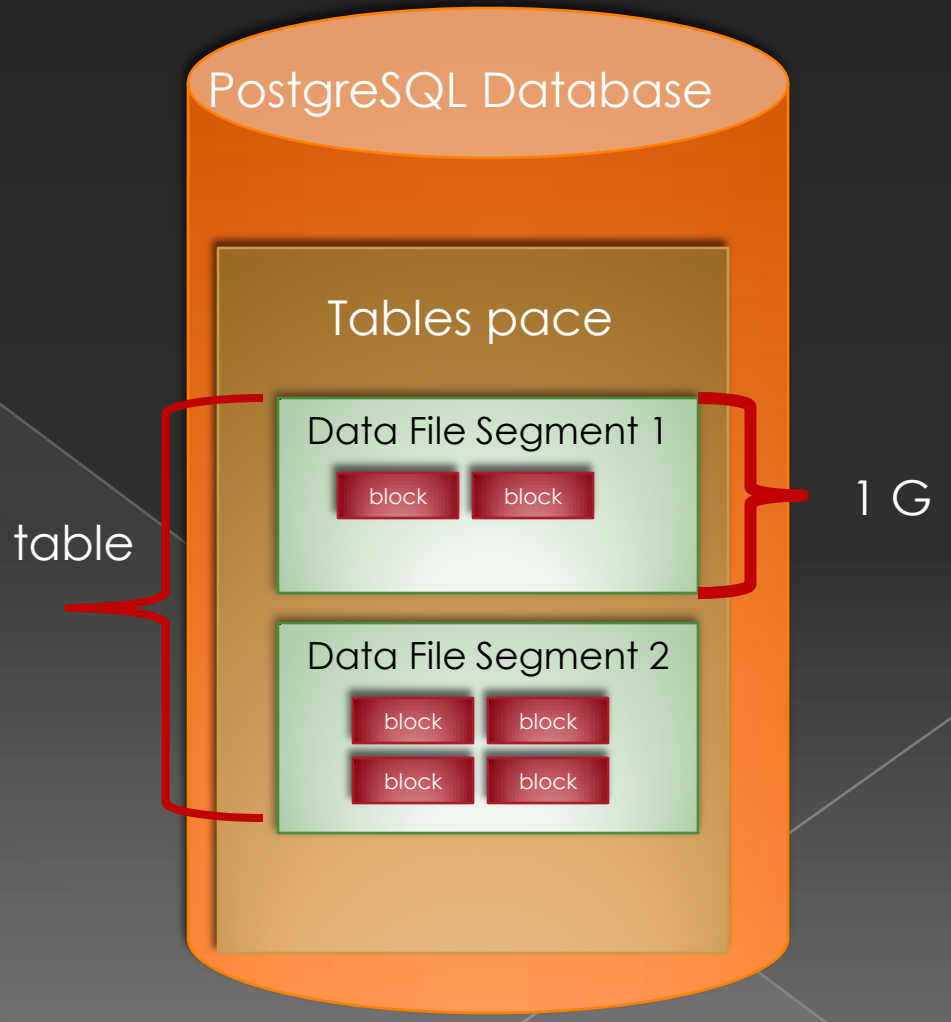
内存的更新关系


Work_mem
postgres
1. Update data
2. Write transaction log
Data Buffer
(shared buffer)
Trx Log Buffer
(WAL buffer)
Data writer
(background writer)
5. checkpoint
3. commit
OS I/O cache
6. fsync
4. fsync
Commit
finished
Data files
WAL files

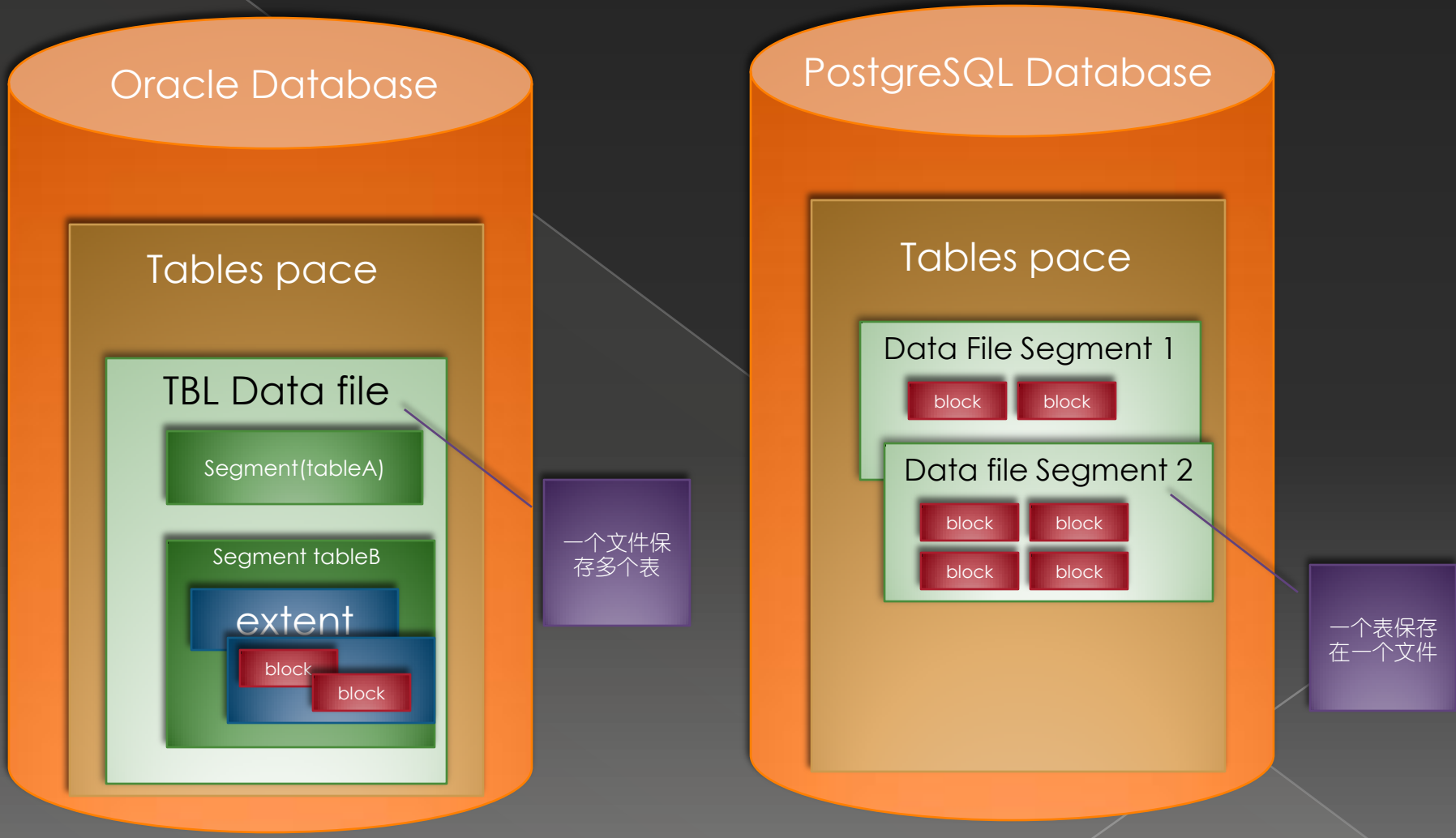
存储结构

存储的结构

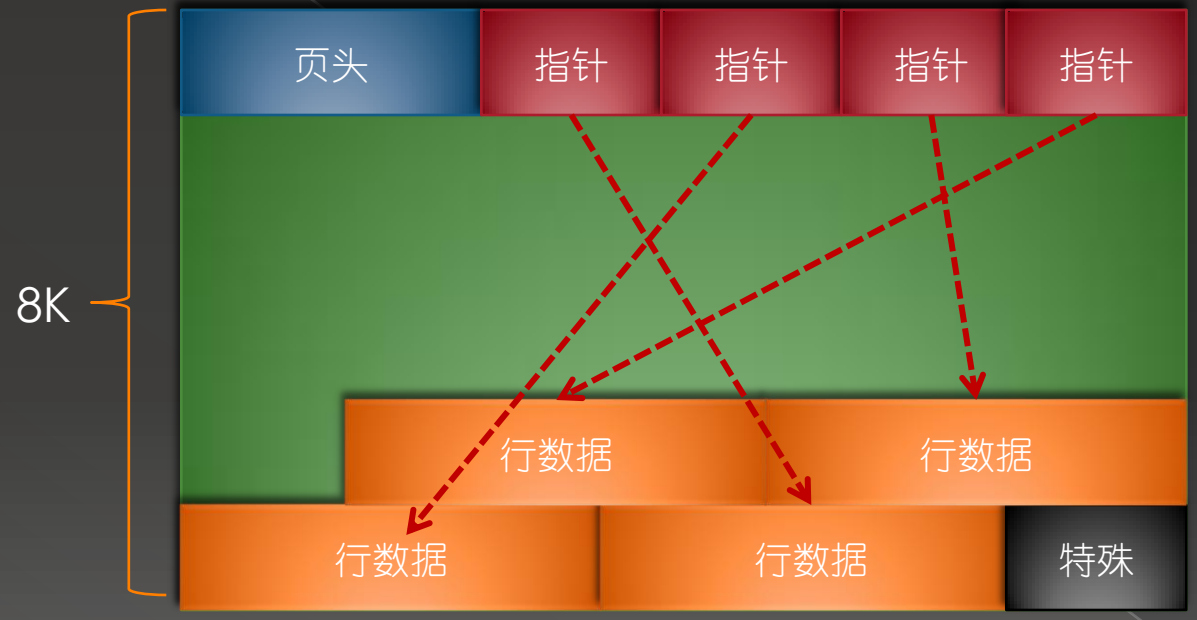
- 表空间(table space)
  - 保存数据文件
- 数据文件(data file)
  - 表存放在一个单独的文件；
  - 如果文件超过1G,将分为多个文件(segment)。
- 块(block)
  - 文件被划分成多个块(8K)



存储的结构比较


Oracle Database
Tables pace
TBL Data file
Segment(tableA)
Segment tableB
extent
block
block
一个文件保存多个表
PostgreSQL Database
Tables pace
Data File Segment 1
block
block
Data file Segment 2
block
block
block
block
一个表保存在一个文件

数据块的结构

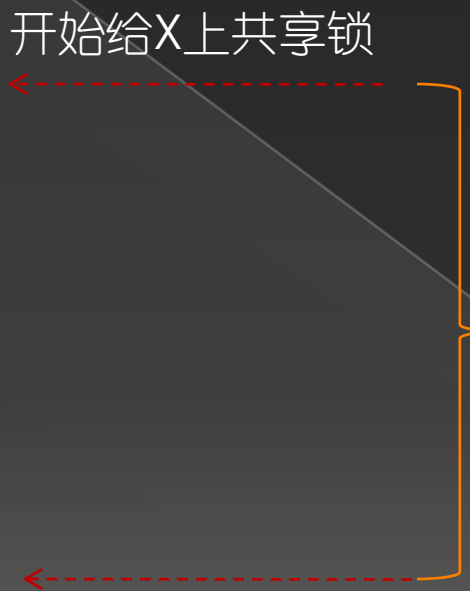


多版本并发控制MVCC

并发控制--封锁及其问题

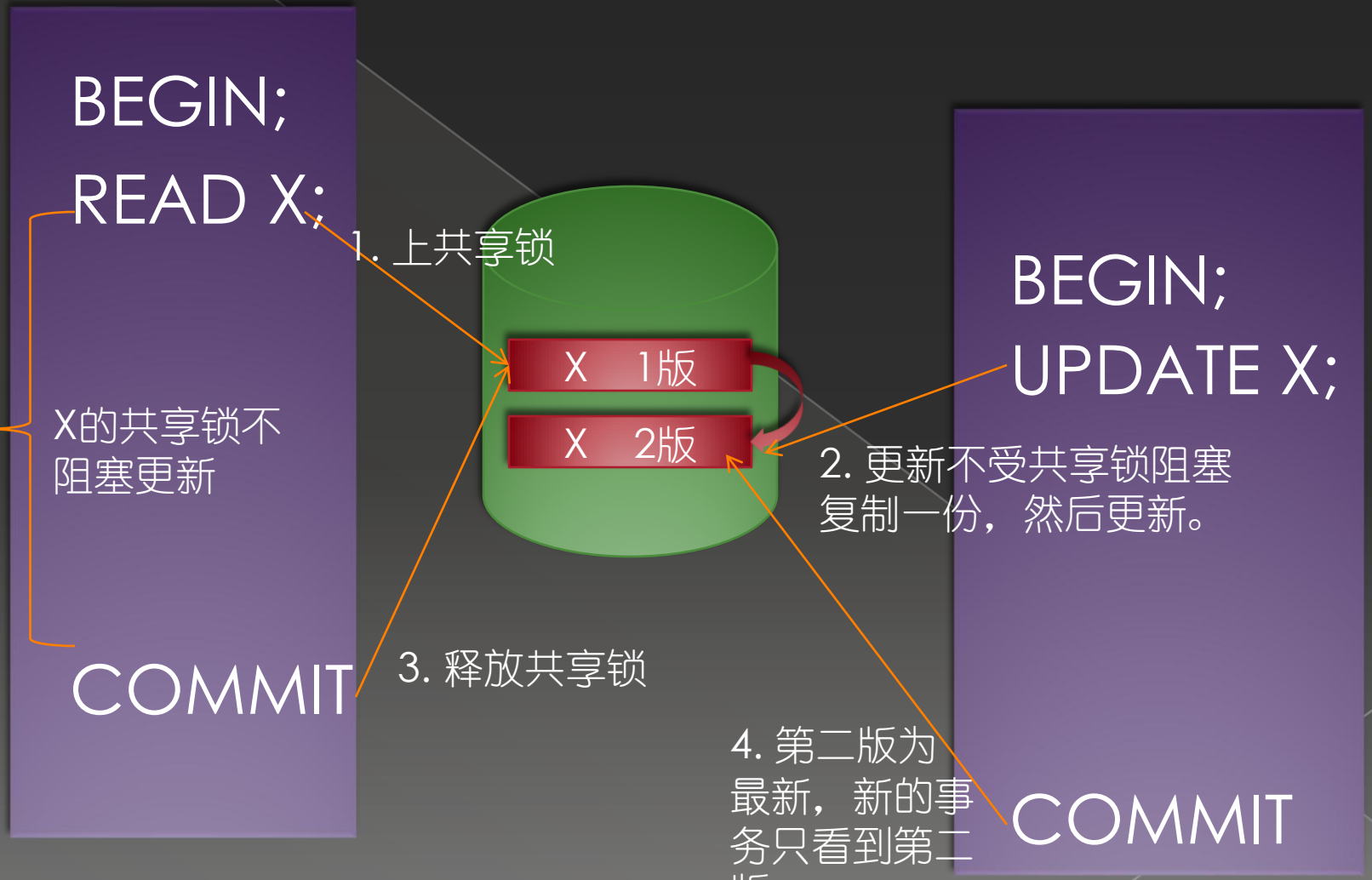
BEGIN;

READ X;

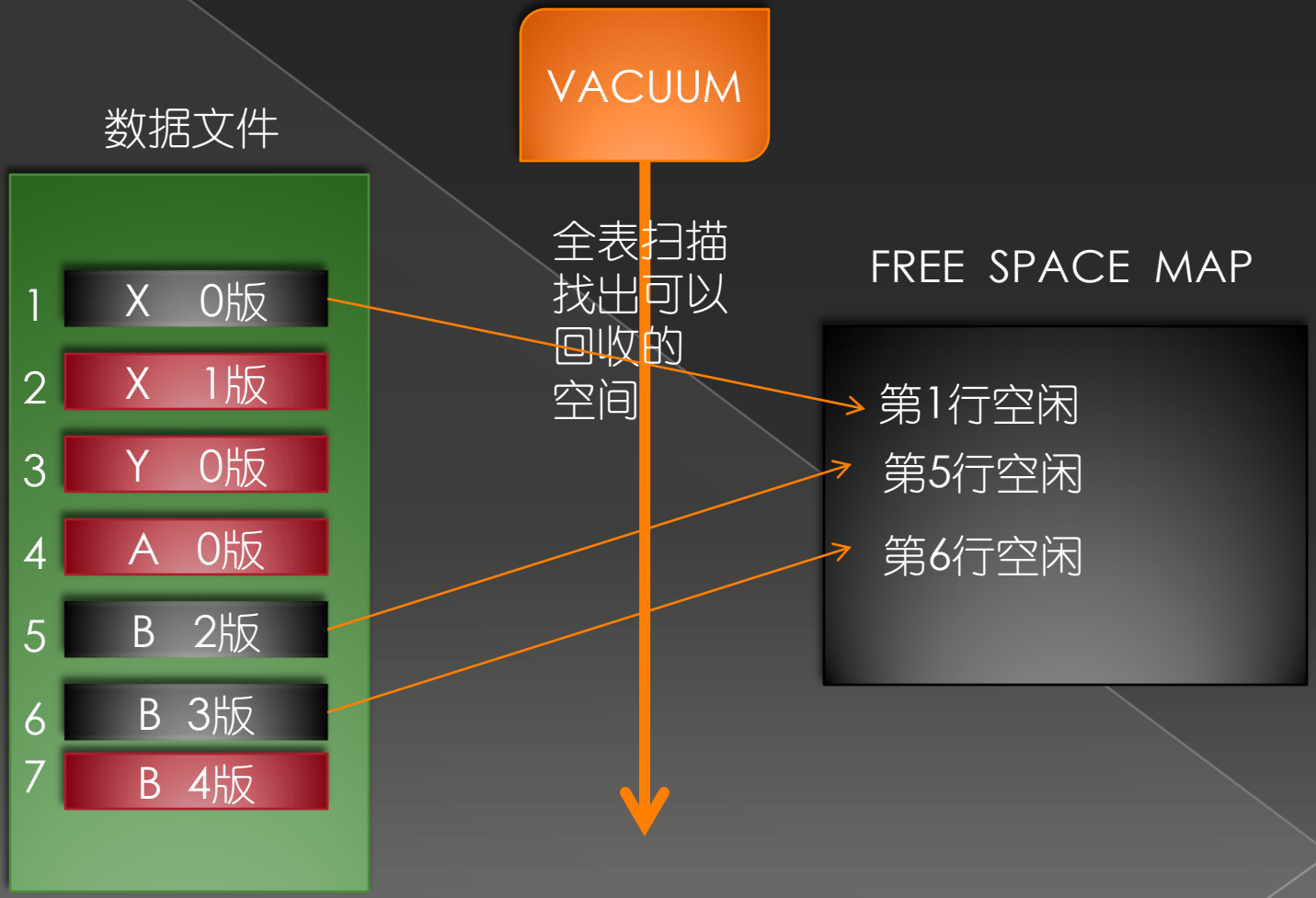


COMMIT

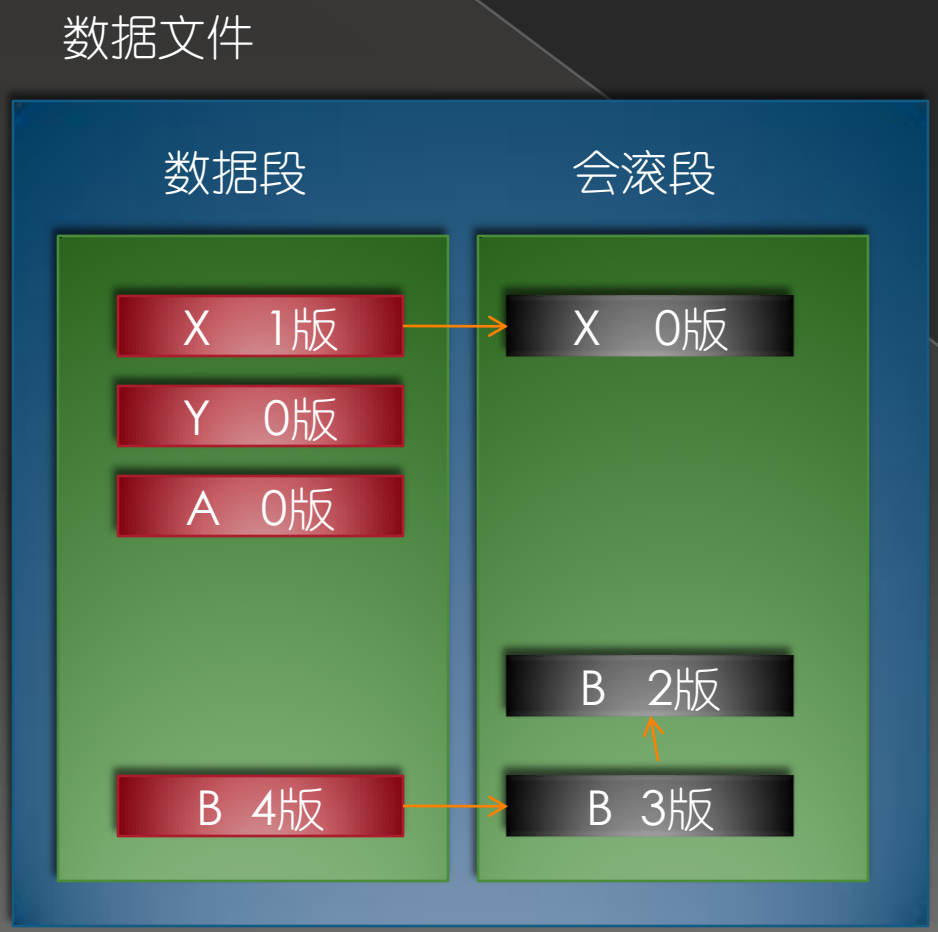
并发控制-多版本控制(MVCC)



空间回收机制-VACUUM



与Oracle多版本机制的对比



Oracle的保存机制

PostgreSQL的保存机制

MVCC机制的优缺点分析

- Oracle
  - 旧版数据与新版数据分别在不同的段，不用进行数据文件扫描回收空间，直接重用会滚段即可；
  - 有‘ora-01555快照太旧’的问题；
  - 需要维护会滚段
- PostgreSQL
  - 旧版数据及新版数据在同一个文件，需要定时扫描文件，回收空闲空间；
  - 不需要维护会滚段；

空间回收机制的改善

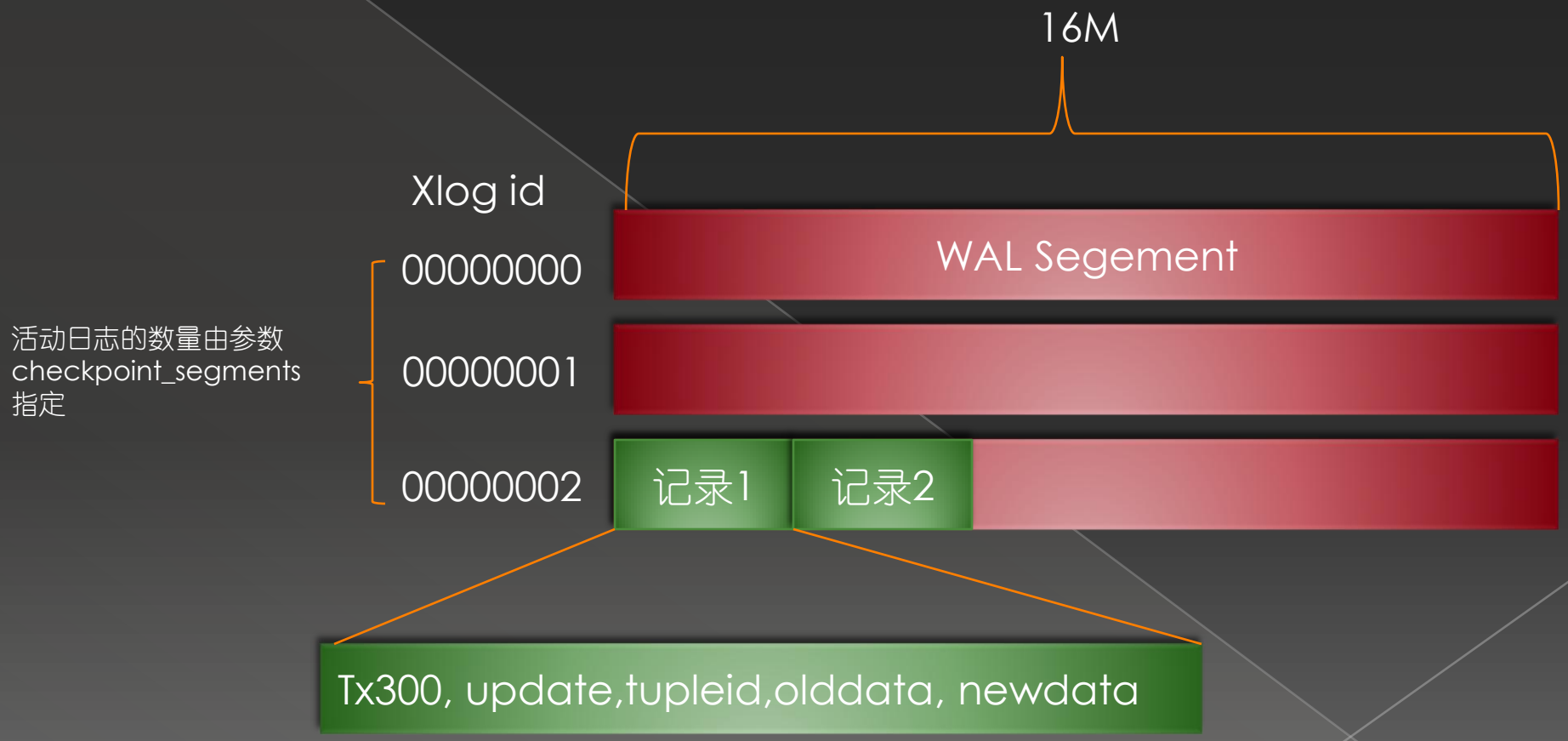
- Concurrency vacuum
- Cost delay vacuum
- Autovacuum
- Dead Space Map

预写式事务日志WAL

预写入事务日志WAL

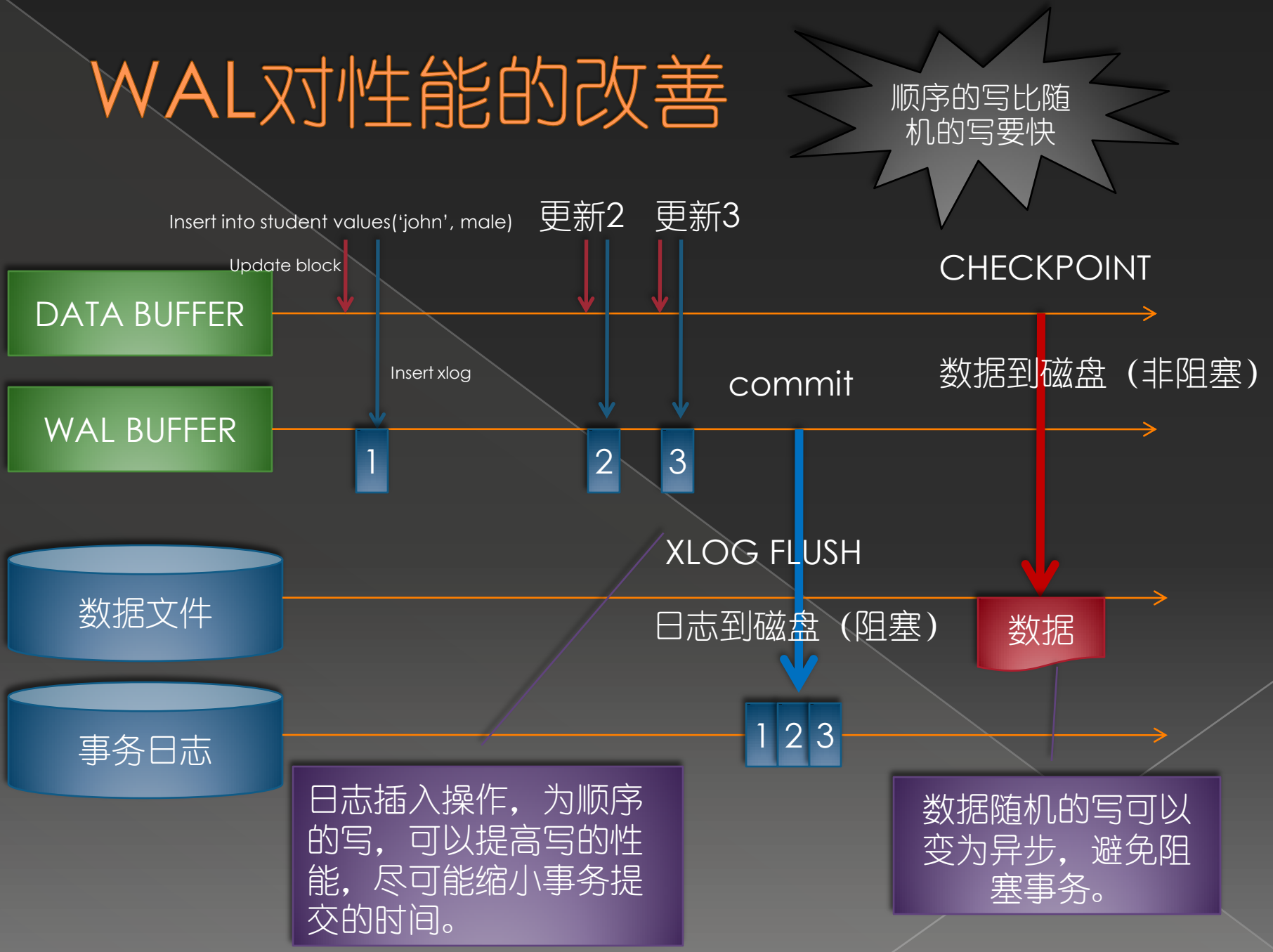
- 什么是WAL
  - 在更新实际的数据之前，把操作的的信息(增/删/改)及数据作为事务日志，写到硬盘保存；
  - 只有日志实际写到硬盘以后，数据才被认为安全；
  - 当数据没有被写到磁盘时候，系统崩溃，可以用日志重新更新数据，确保事务的原子性。
- 与Oracle的REDO LOG相似

WAL的文件结构

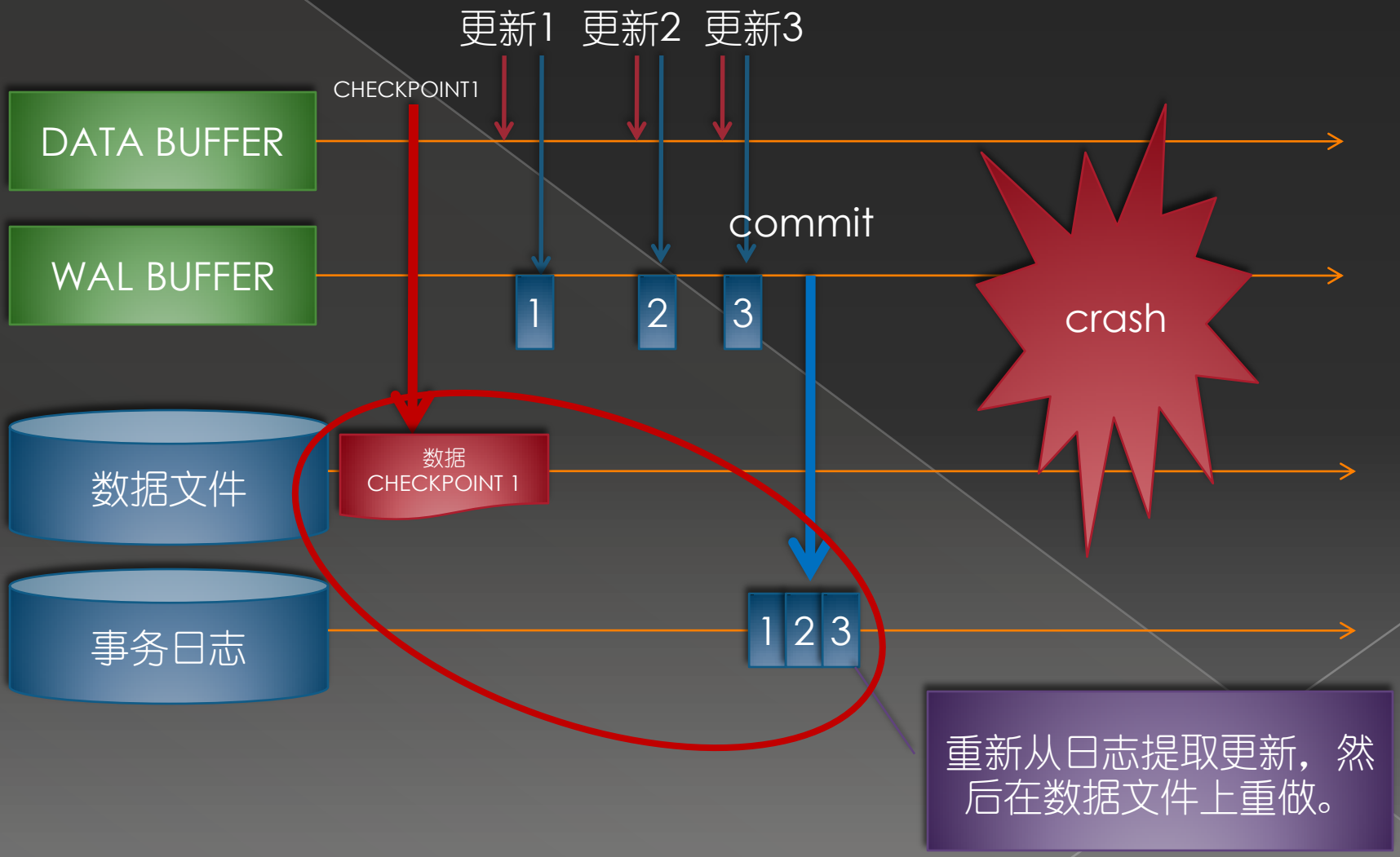


更新操作的事务日志

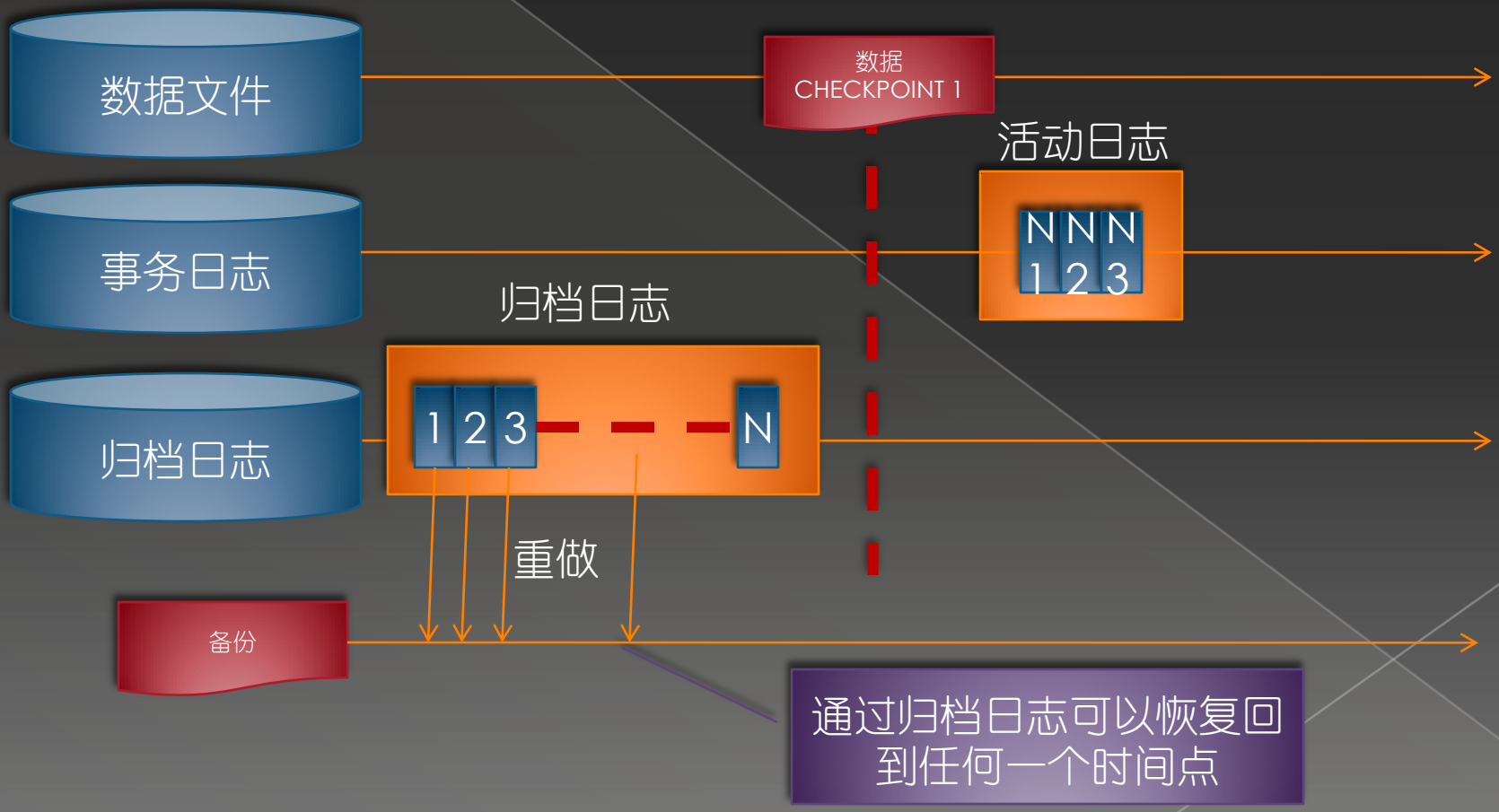
WAL对性能的改善


顺序的写比随机的写要快
Insert into student values('john', male)
更新2
更新3
Update block
CHECKPOINT
DATA BUFFER
Insert xlog
commit
数据到磁盘（非阻塞）
WAL BUFFER
1
2
3
XLOG FLUSH
数据文件
日志到磁盘（阻塞）
数据
事务日志
1 2 3
日志插入操作，为顺序的写，可以提高写的性能，尽可能缩小事务提交的时间。
数据随机的写可以变为异步，避免阻塞事务。

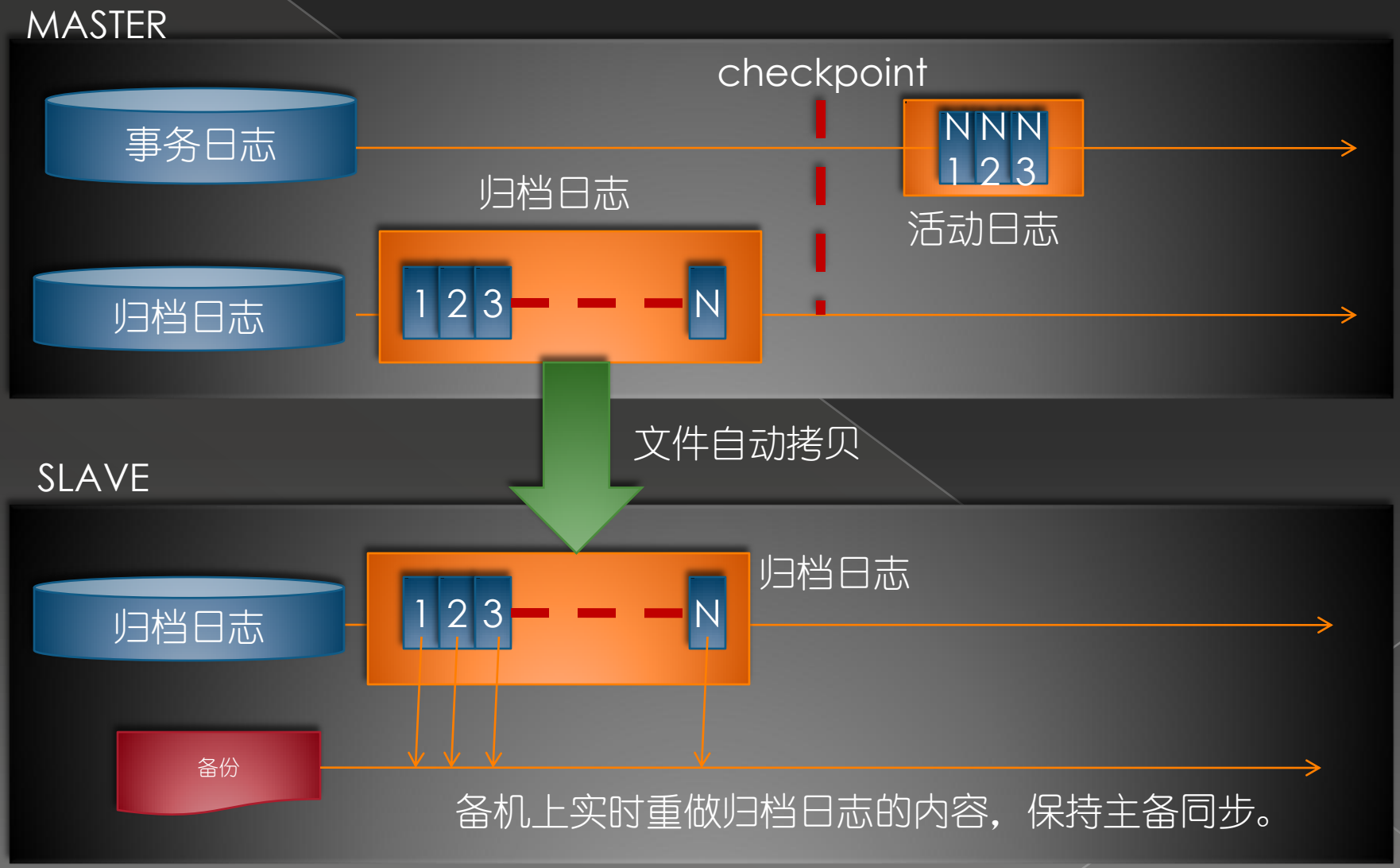
事务日志-Crash Recovery



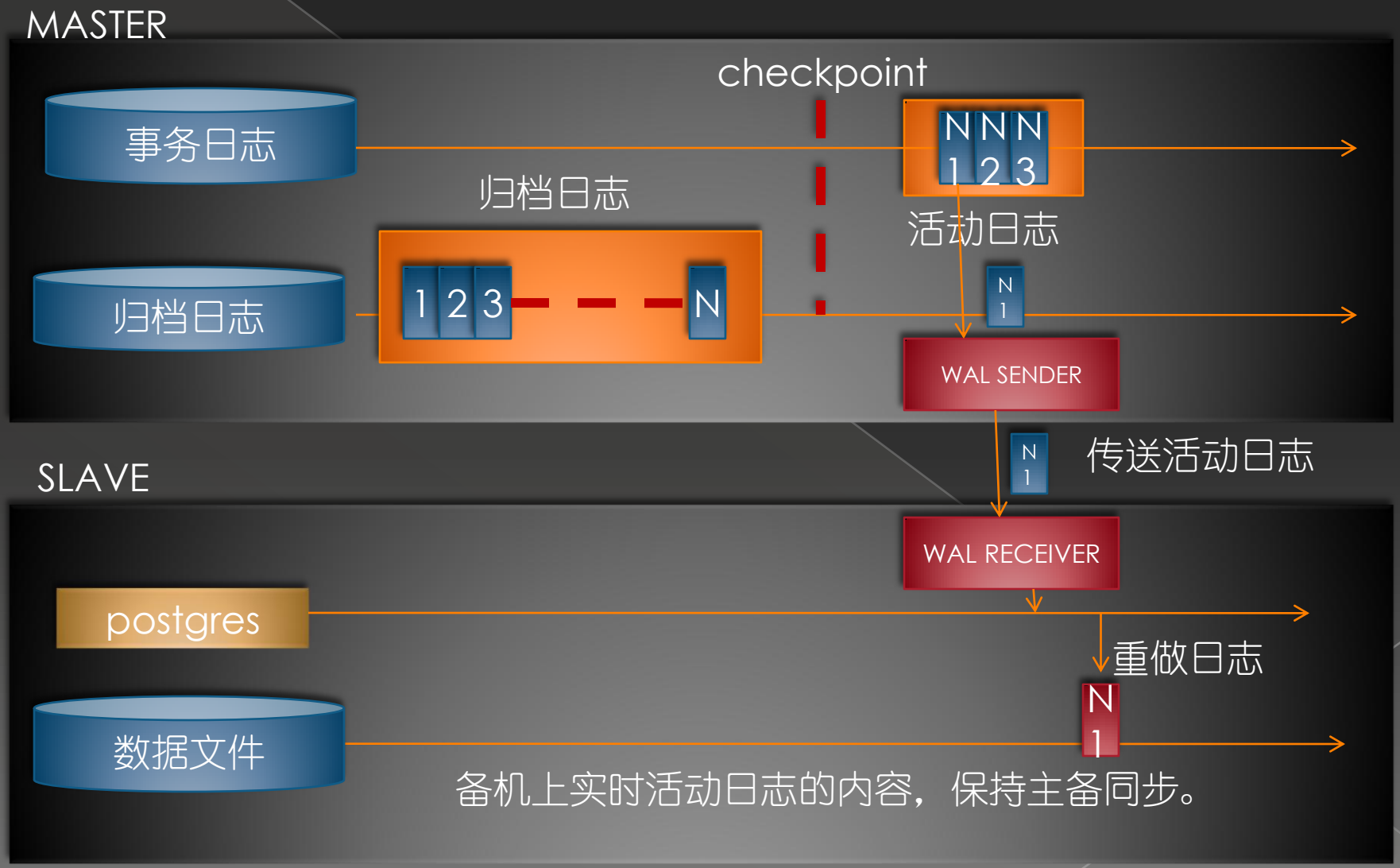
事务日志-PITR


数据文件
事务日志
归档日志
数据
CHECKPOINT 1
活动日志
N N N
1 2 3
归档日志
1 2 3
N
重做
备份
通过归档日志可以恢复回
到任何一个时间点

事务日志-WARM STANDBY

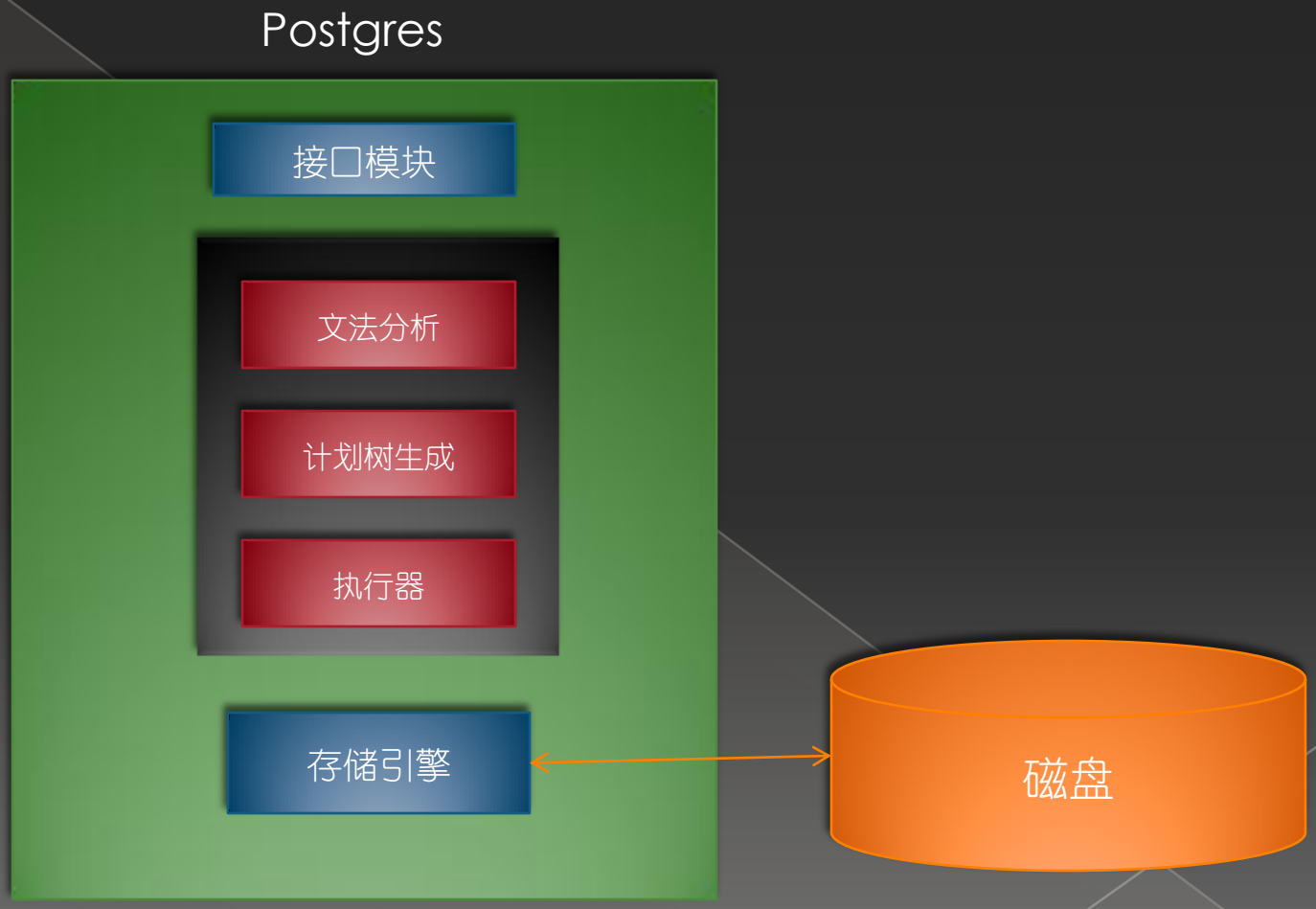


事务日志-HOT STANDBY



postgres架构

Postgres的体系结构



谢谢

李元佳

galylee@gmail.com 13710389225